

Using Domain-Specific Modeling for Design and Verification of Cyber Physical Systems

17 March, 2016

Juha-Pekka Tolvanen
jpt@metacase.com

Motivation

- Domain expert vs. verification expert
 - Domain experts are not necessarily familiar with development and testing tools
 - Developers and verification experts don't always master the domain knowledge
- Domain-specific languages enable to use directly the domain concepts for both development and V&V
 - Common vocabulary to enable feedback and communication
 - Higher level languages improve productivity
 - Automated transformations improve quality

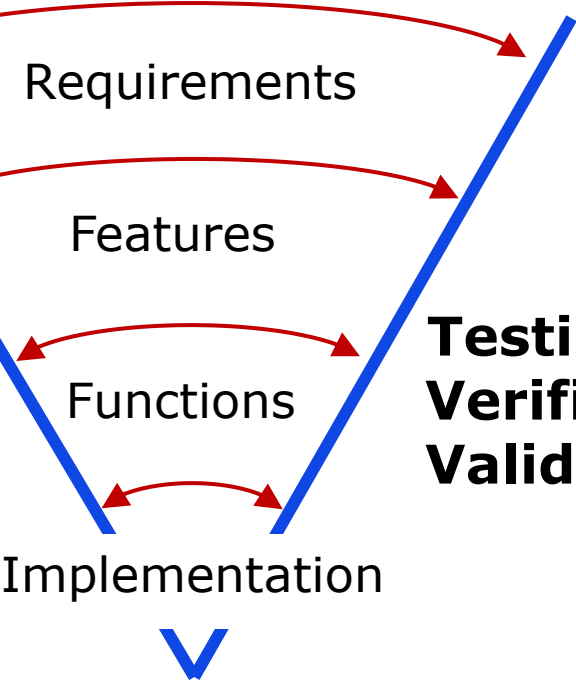
Separate vs. common language

**Domain knowledge
(subject matter expert)**



**Technical knowledge
(developers)**

Development



**Testing
Verification
Validation**

Domain-Specific Modeling (DSM)

- Use of concepts from the problem domain
 - Already familiar => no need to learn new
 - Have known semantics
- Having a special focus
 - Use concepts that are relevant for the task: development, testing, verification, validation
- Use concrete syntax that enables communication and collaboration close to domain's natural representation
 - Not a cryptic programming/scripting language
- DSM is applied in particular for automating repetitive development efforts*, but less in testing and V&V

* See references on EADS, NSN, Nokia, Panasonic, Polar, Elektrobit, USAF

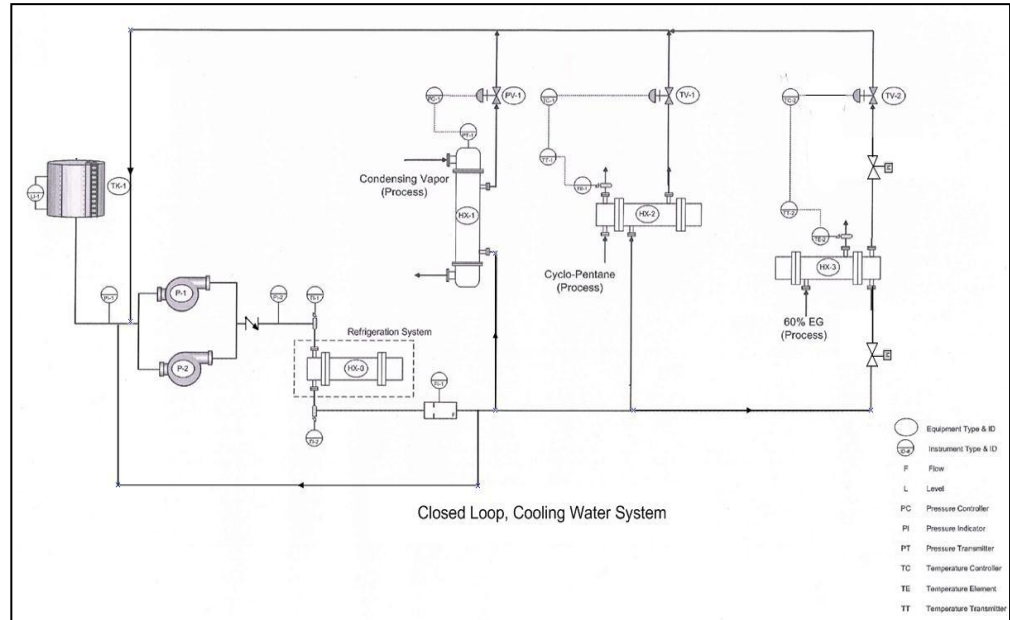
Example: Industrial Process Plant



Example Specification



Closed loop, Heat transfer, Liquid circulating (CHL)



May include:

- System Requirements Tree
- System Requirements
- Component Requirements
- Interface Requirements

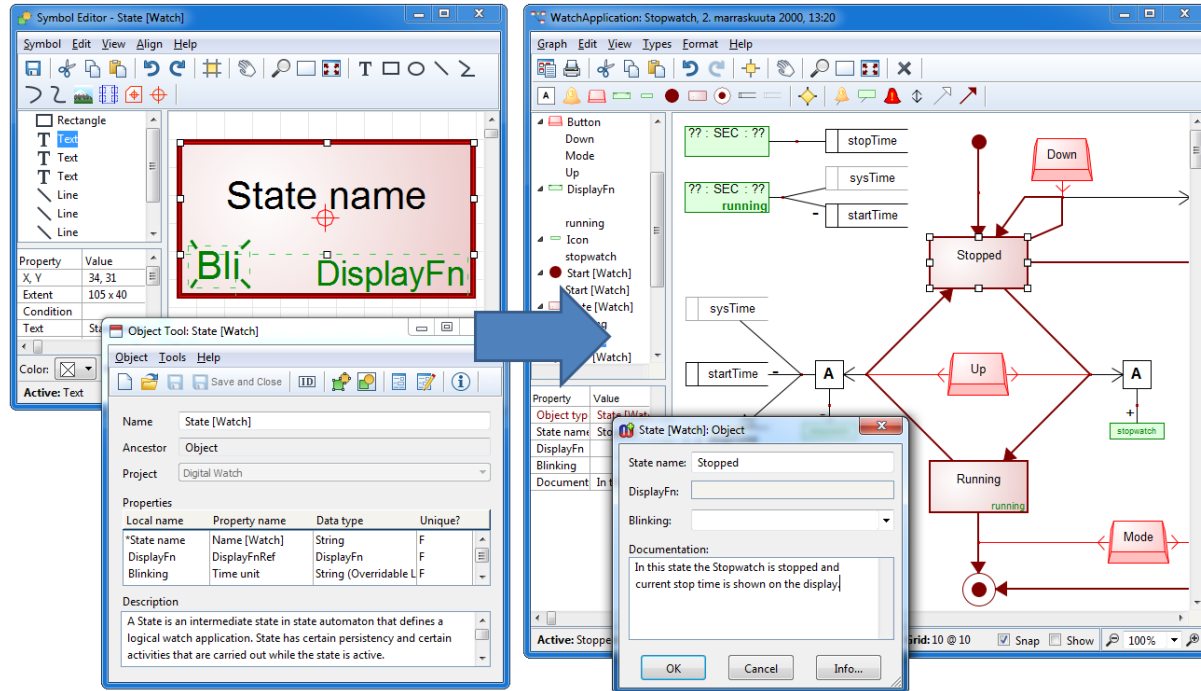
How to define a language for a given domain: steps

1. Identify abstractions
 - Concepts and how they work together
 2. Specify the metamodel
 - Language concepts and their rules
 3. Create the notation
 - Representation of models
 4. Define the generators
 - Various outputs and analysis of the models
- The process is iterative: try solution with examples
- Define part of language, model with it, define more...

Roles for language definition & use

Experts define languages & generators

Team models with domain concepts & generate code, tests...



Step 1) Identify abstractions from domain terminology

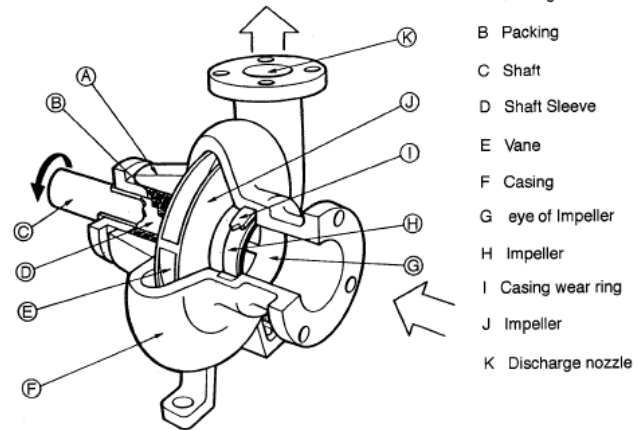
- Detailed information specifying functional & physical characteristics of a component of a system, plant or facility (e.g. pump)

CENTRIFUGAL PUMP API-610
DATA SHEET
MKS UNITS

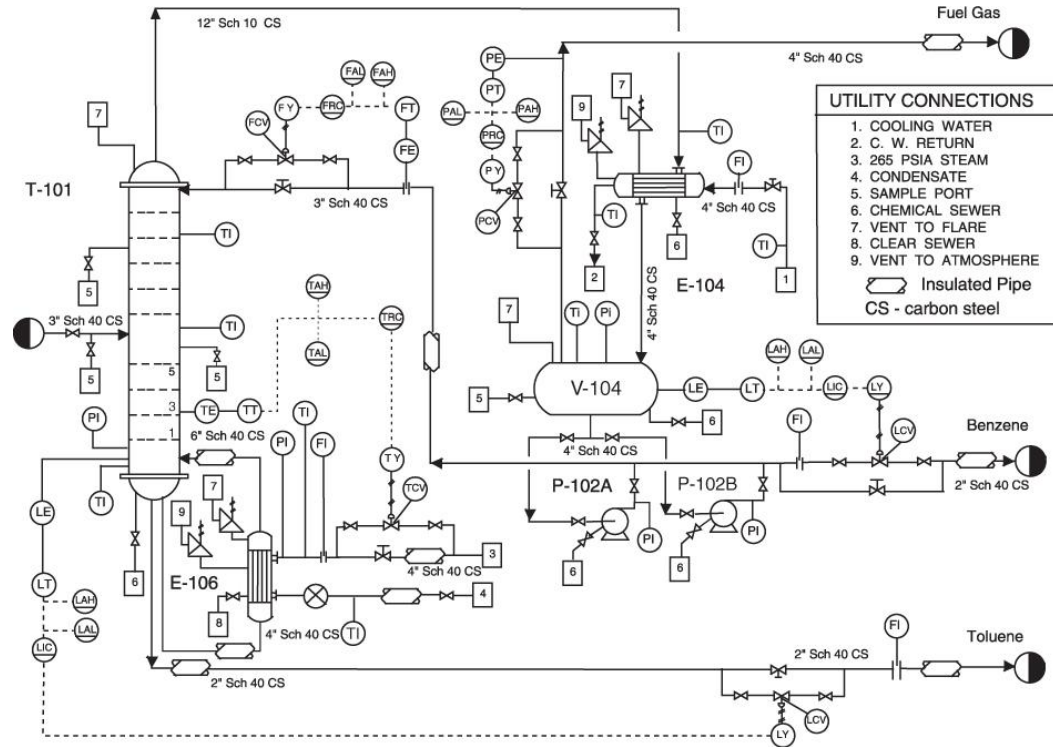
JOB NO. _____ ITEM NO. _____
REQ / SPEC No. _____ / _____
PURCH ORDER No. _____ DATE 29-Oct-09
INQUIRY No. _____ BY DAA

1 APPLICABLE TO: PROPOSAL PURCHASE AS BUILT
2 FOR: _____ UNIT: _____
3 SITE: _____ SERVICE: _____
4 NO. REQ. _____
5 MANUFACTURER _____
6 NOTES: INFORMATION _____
7 _____
8 PUMPS TO OPERATE IN _____
9 (SERIES) WITH _____
10 GEAR ITEM No. _____
11 GEAR PROVIDED BY _____
12 GEAR MOUNTED BY _____
13 GEAR DATA SHT. No. _____
14 **OPE**
15 CAPACITY NORMA _____
16 OTHER _____
17 SUCTION PRESSUR _____ g/cm²
18 DISCHARGE PRESS _____
19 DIFFERENTIAL PRES _____
20 DIFF. HEAD _____
21 PROCESS VARIATI _____
22 STARTING CONDITIONS _____ VAPOR PRESSURE _____ (Kg/cm²) _____ (°C)
23 SERVICE: CONT. INTERMITTENT (STARTS/DAY) _____ RELATIVE DENSITY (SPECIFIC GRAVITY): _____
24 PARALLEL OPERATION REQ'D _____ NORMAL _____ MAX _____ MIN _____

Product Data Sheet

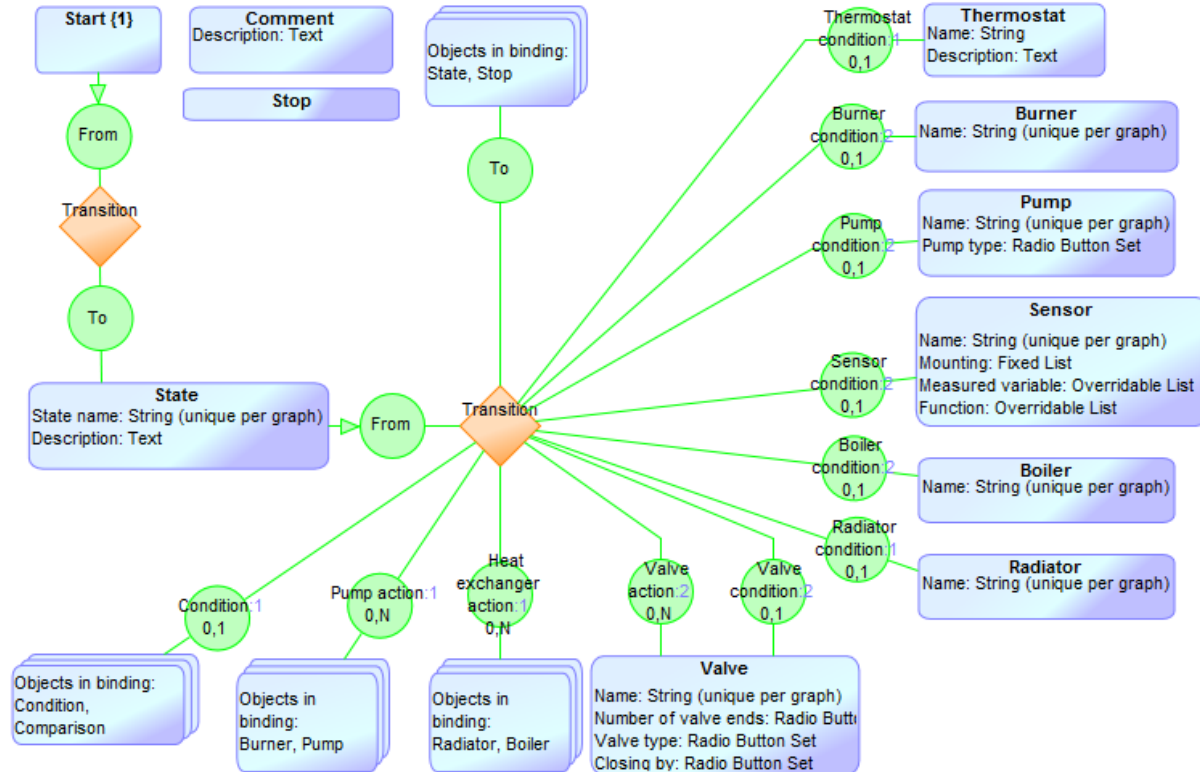


Design with domain-concepts & abstractions



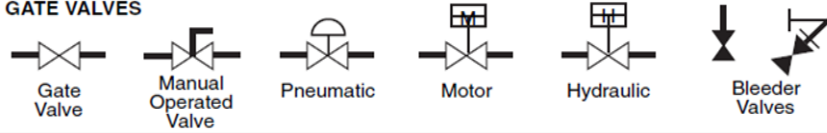
* Turton et al., Analysis, Synthesis and Design of Chemical Processes, Prentice Hall. 2012

Step 2) Specify the metamodel

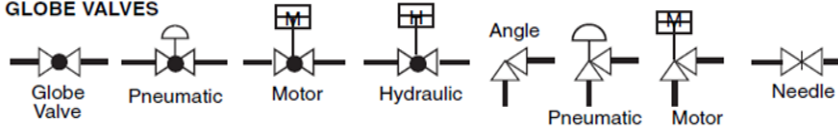


Domain terminology and visualization: Valves

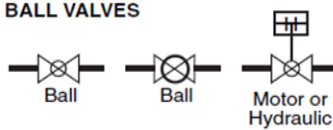
GATE VALVES



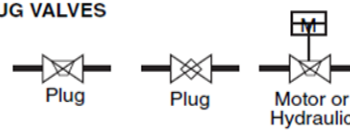
GLOBE VALVES



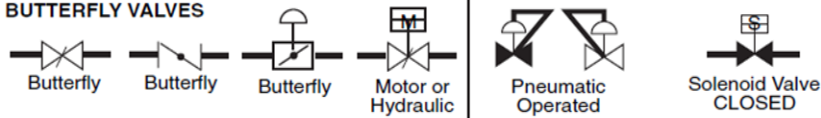
BALL VALVES



PLUG VALVES



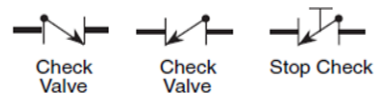
BUTTERFLY VALVES



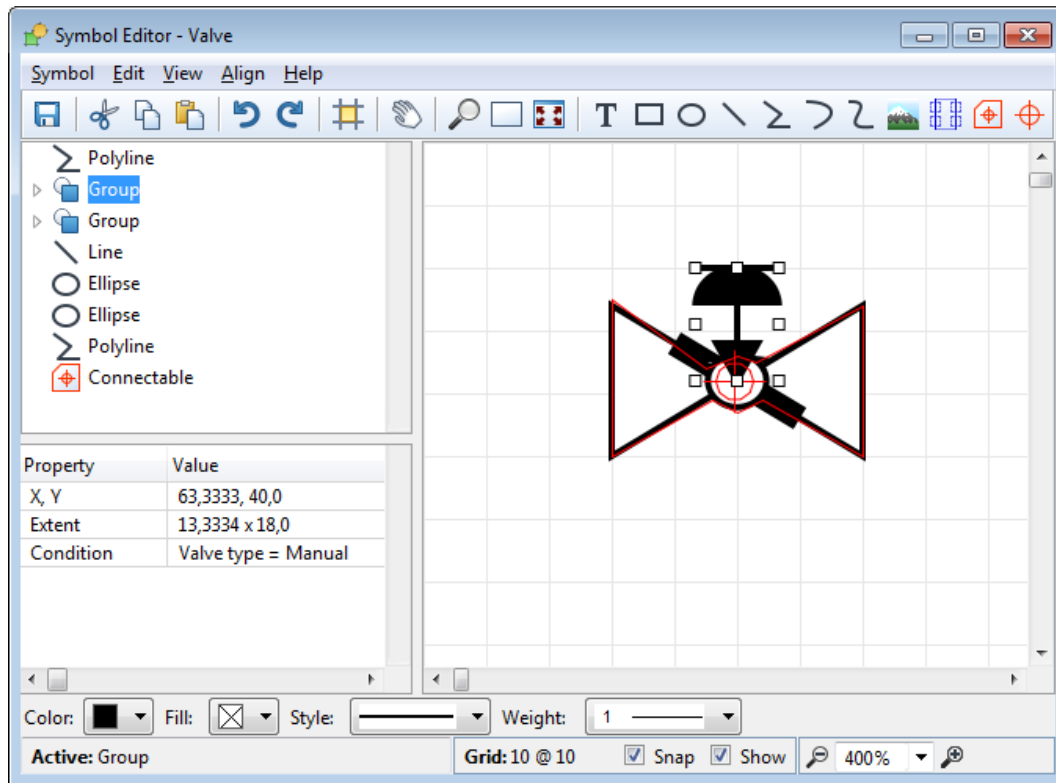
DIAPHRAGM VALVES



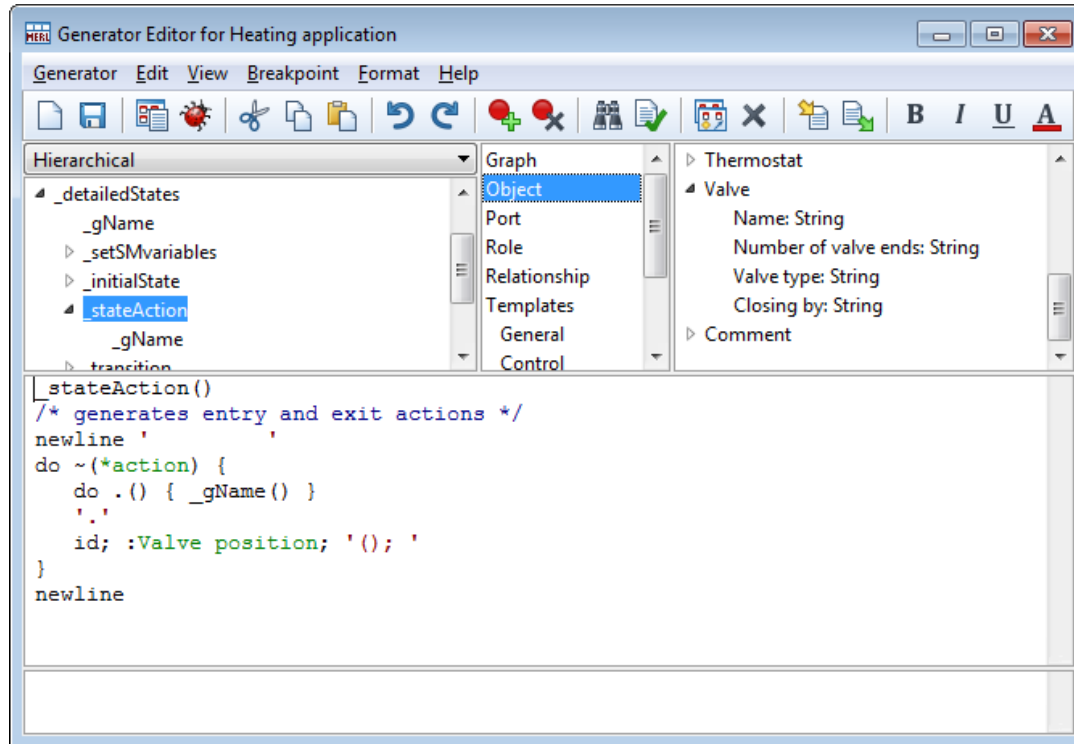
CHECK VALVES



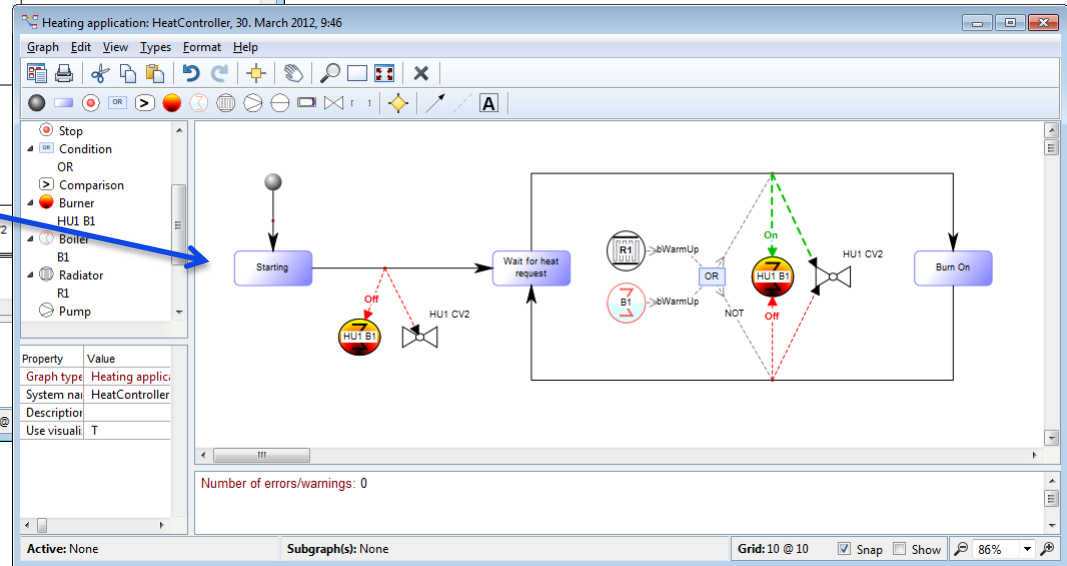
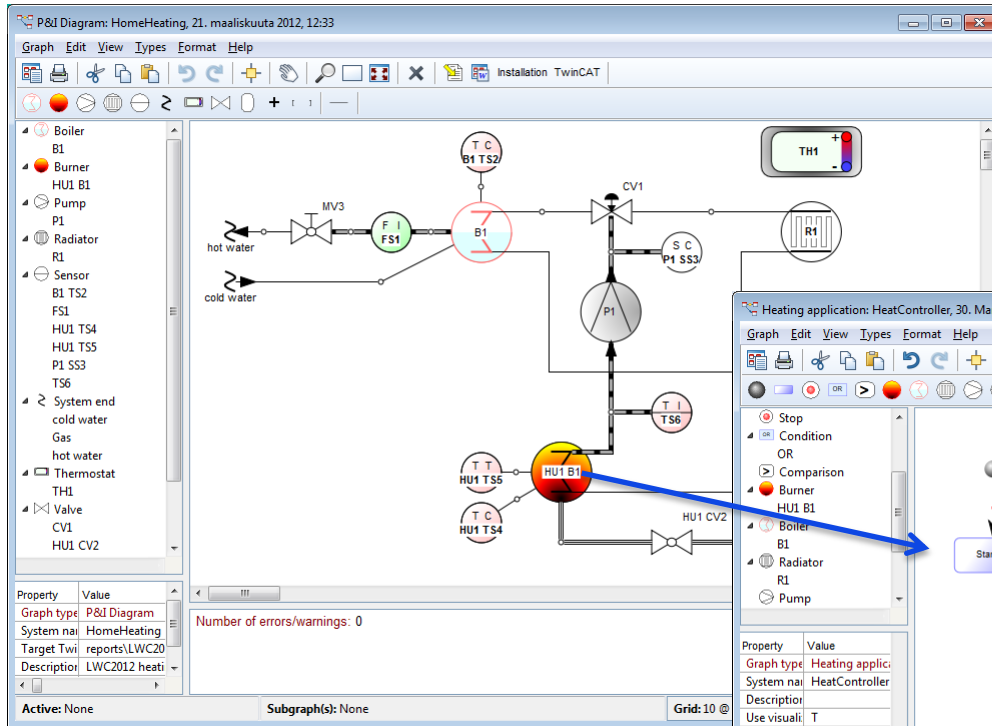
Step 3) Create the notation: Example on Valve



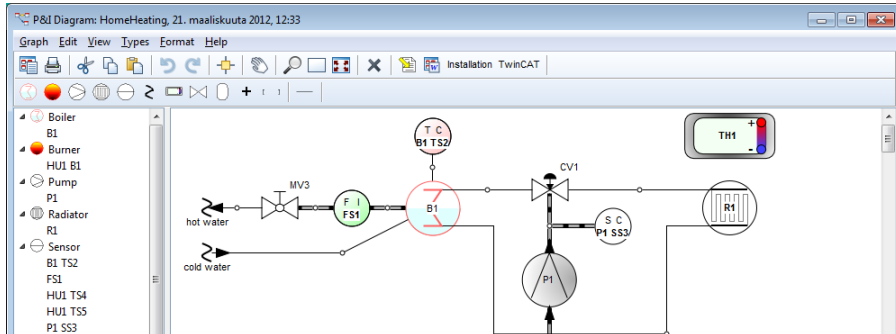
Step 4) Define the generators: PLC code for Valve actions



Developing the system with DSM



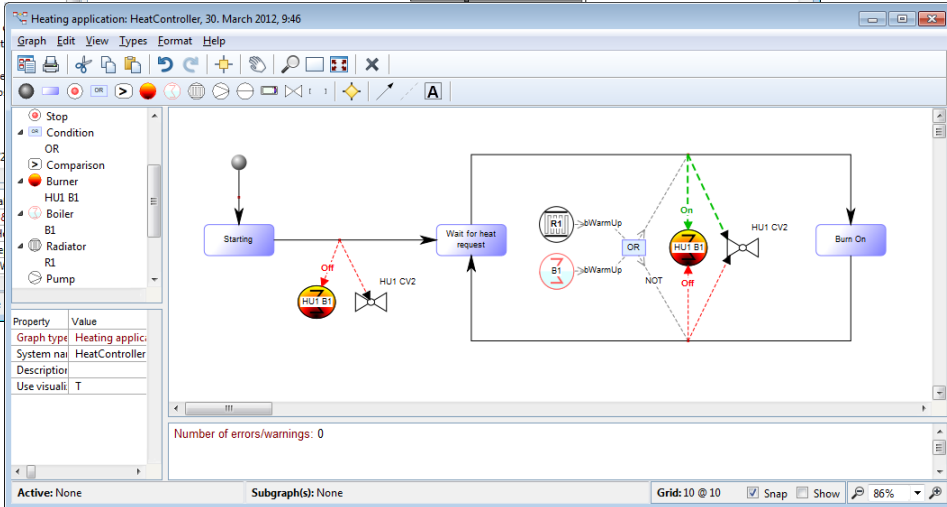
Generating the code from models



PLC code

TwinCAT PLC Control - LWC2012_HomeHeatingPlatform.pro* - [MDL_PumpController_SM_rg (ST) - FB_HomeHeating (FB-S...]

```
00001 * PumpController state machine generated from MetaEdit+*
00002 (* Pump is put on or off depending on flame detection. *)
00003
00004 MDL_PumpController_SM_bIsEntry := MDL_PumpController_SM_bATransition
00005
00006 IF MDL_PumpController_SM_bIsEntry THEN
00007     MDL_PumpController_SM_eLastState := MDL_PumpController_SM_
00008     MDL_PumpController_SM_bATransitionWasPerformed := FALSE;
00009 END_IF
00010
00011 CASE MDL_PumpController_SM_eCurrentState OF
00012
00013     MDL_PumpController_SM_Initial:
00014
00015     IF NOT MDL_PumpController_SM_bATransitionWasPerformed THEN
00016         MDL_PumpController_SM_eCurrentState := MDL_PumpCo
00017         MDL_PumpController_SM_bATransitionWasPerformed :=
00018     END_IF
00019
00020 MDL_PumpController_SM_CONTROL_PUMP:
00021
00022     IF NOT MDL_PumpController_SM_bATransitionWasPerformed THEN
00023         IF MDL_HU1_B1_bFlameDetected THEN
00024             MDL_PumpController_SM_eCurrentState := MDL_PumpCo
00025             MDL_PumpController_SM_bATransitionWasPerformed :=
00026                 MDL_P1_On();
00027         END_IF
00028     END_IF
00029
00030 IF NOT MDL_PumpController_SM_bATransitionWasPerformed THEN
```



How to test a cooling system?

1. Design time

- Domain-Specific Modeling Language already captures several rules of the system
- Language prevents errors already in the design stage

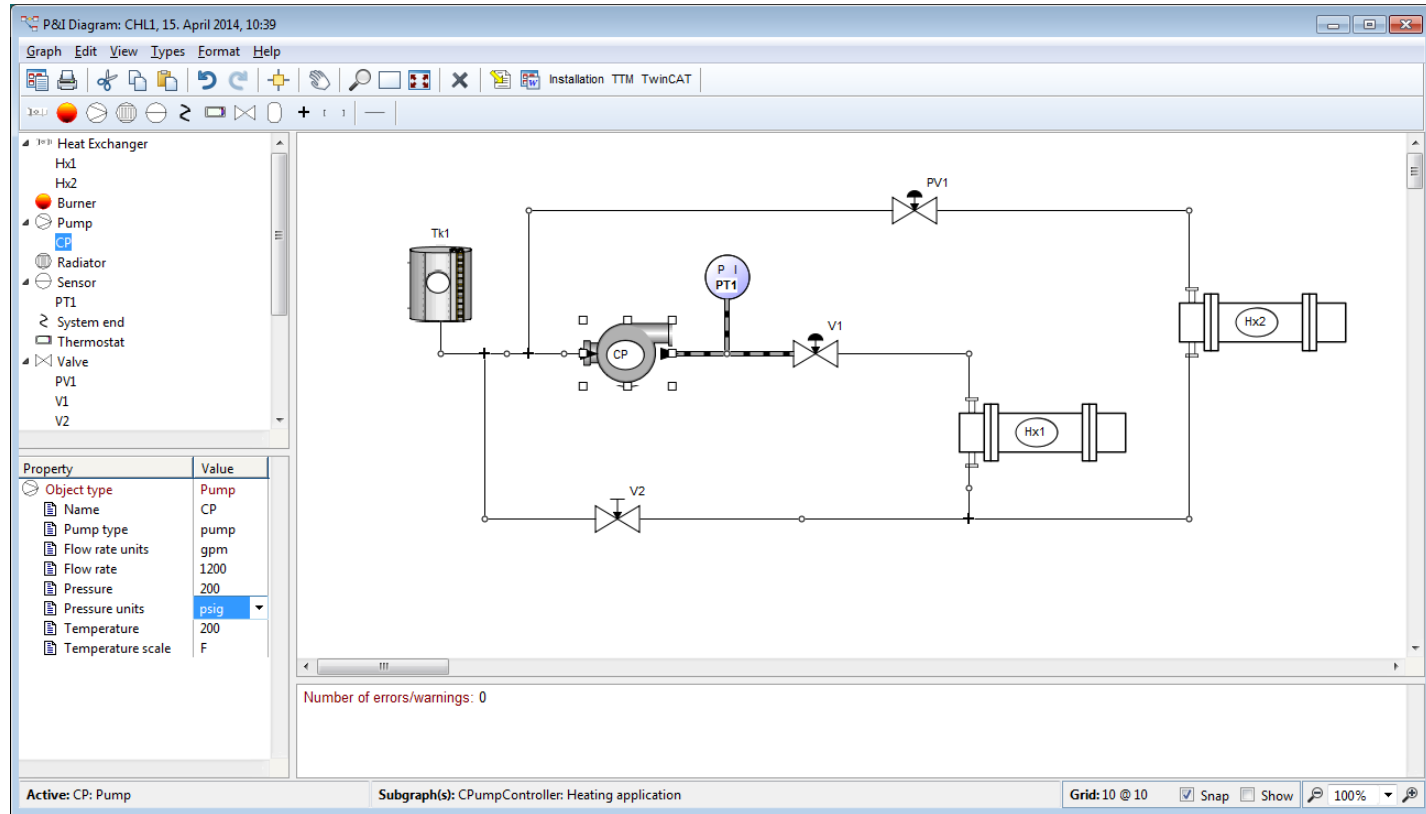
2. Testing and V&V

- DSL can capture aspects related to testing and V&V
- Same language concepts used for both development as well as for testing

How to test a cooling system?

- As components (e.g., pumps, valves, heat exchanger) wear out, new components are substituted
 - Common for original requirements or design to not exist
 - May not know how current facility implementation deviates from original design or requirements
- Concern: newly substituted component can create potential operational or safety issues such as:
 - Temperature: Produce too much heat?
 - Pressure: Incorrect input/output pressure?
 - Flow rates: Conflicting flow rates in the configuration?
 - Control logic errors...
 - Instrument configuration...

Example: Cooling in process plant*



* M. Blackburn, P. Denno, Virtual Design and Verification of Cyber-Physical Systems

Specifying properties of components

The screenshot displays the P&ID Diagram software interface for a process titled "CHL1, 15. huhtikuuta 2013, 10:39". The main window shows a process flow diagram with components: Tank (Tk1), Pump (CP), Pressure Transmitter (PT1), Valve (V1), Valve (PV1), and Heat Exchangers (Hx1, Hx2). Three dialog boxes are open to specify component properties:

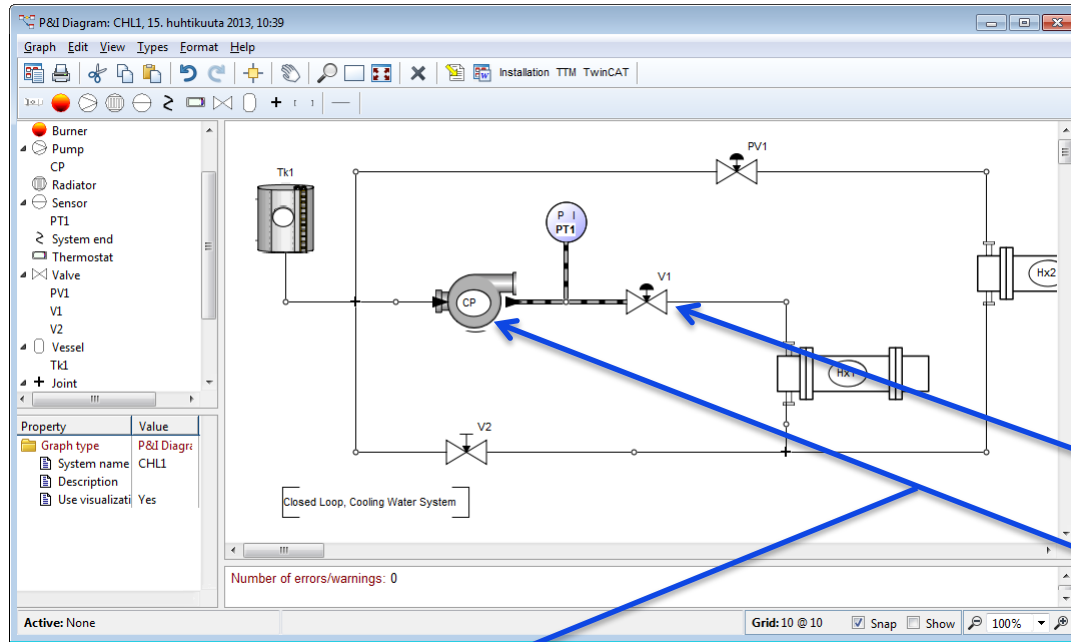
- Valve: Object** (for V1):
 - Name: V1
 - Number of valve ends: Two
 - Valve type: Normal, Manual, Control
 - Closing by: ball, butterfly, diaphragm, needle
 - Flow rate: 1200
 - Flow rate units: gpm
- Heat Exchanger: Object** (for Hx1):
 - Name: Hx1
 - Flow rate: 1200
 - Flow rate units: gpm
 - Pressure: 200
 - Pressure units: psig
 - Temperature: 200
 - Temperature scale: F
- Pipe: Relationship** (for the pipe connecting Hx1 and Hx2):
 - Diameter: 10
 - Length: 10
 - Cover: **none** (selected in the dropdown menu)
 - Other options: none, thermally insulated, jacketed, cooled or heated

The software interface includes a menu bar (Graph, Edit, View, Types, Format, Help), a toolbar, and a component library on the left. A property table is visible at the bottom left:

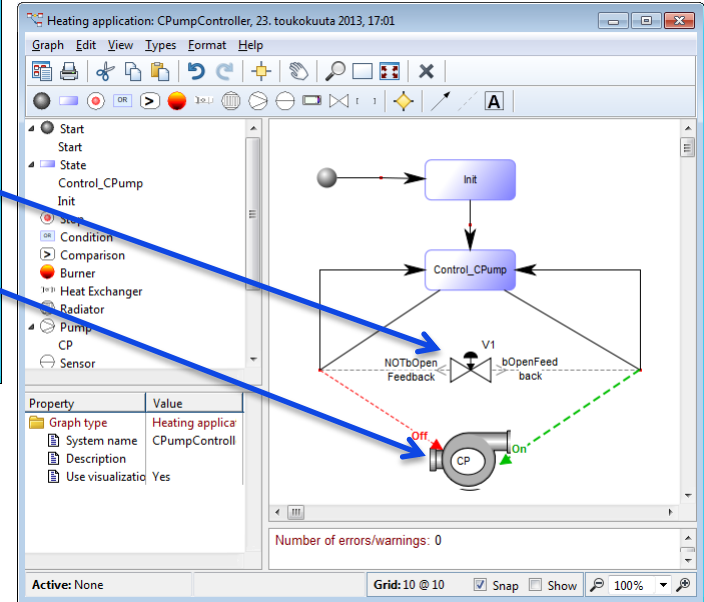
Property	Value
Graph type	P&ID Diagram
System name	CHL1
Description	
Use visualization	Yes

Active: None

Both structure and behavior



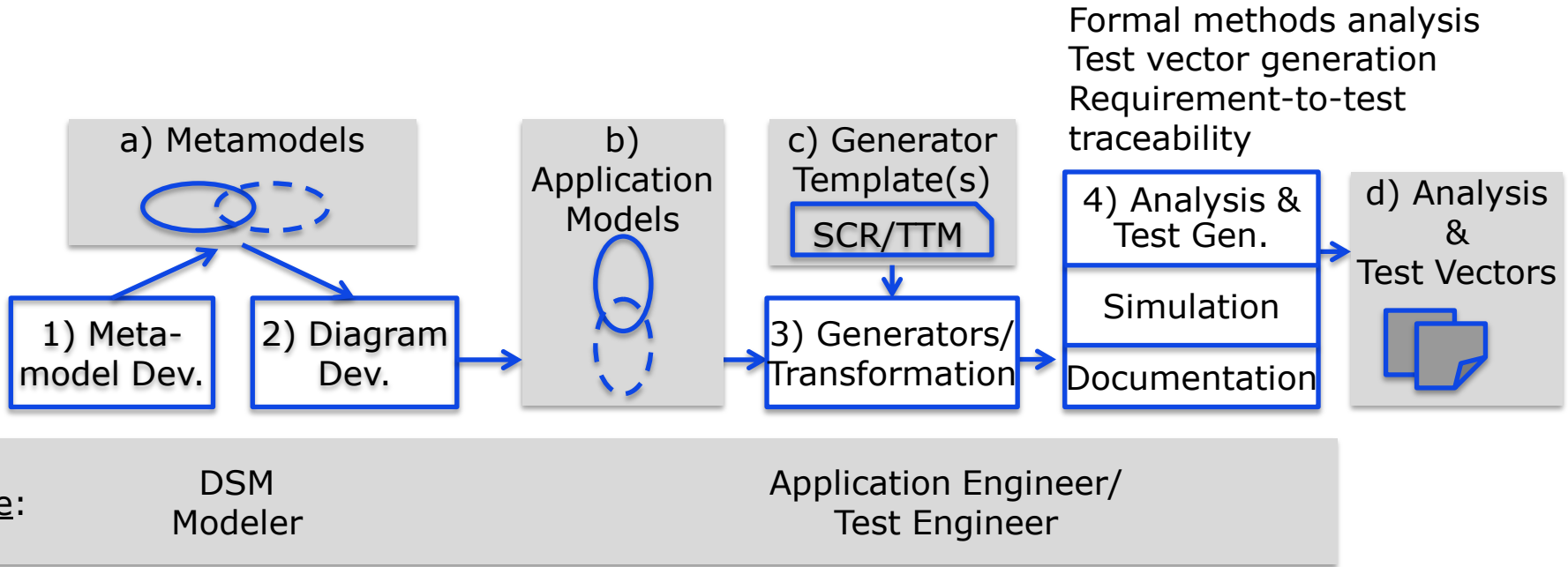
Behavioral constraint:
if valve is closed then
pump should be closed
else if valve is open then
pump can be open



- Same objects: different views used to formalize different aspects of the system
- Languages integrated: can share objects used in different diagram types

Toolchain and stakeholder roles

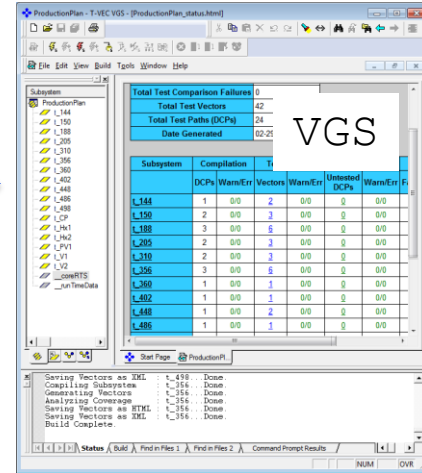
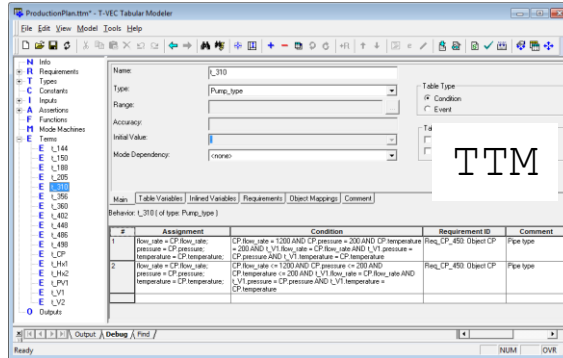
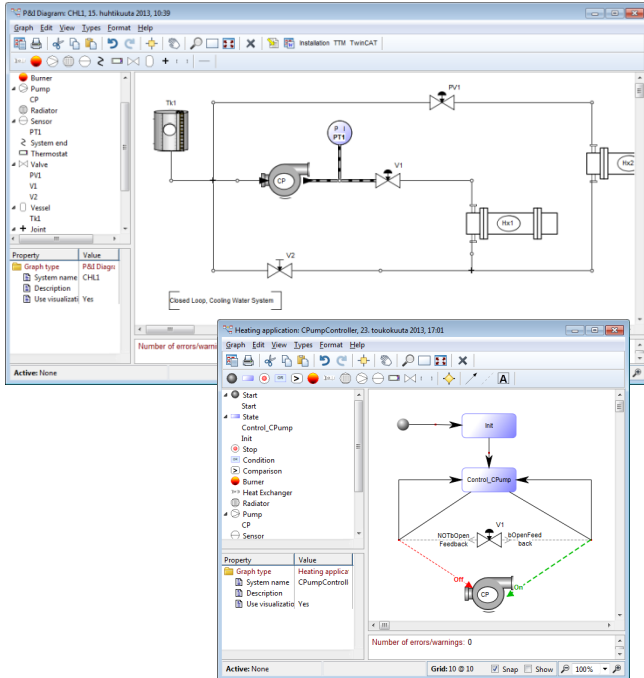
- Conceptual representation of DSM and formal methods toolchain



DSM integrates with formal methods

MetaEdit+ -based DSM
for Process Facility Design

T-VEC Tabular Modeler (TTM) and
T-VEC Vector Generation System (VGS)



- Model transformation
- Formal methods analysis
 - Theorem proving
 - Property checking
- Test vector generation
- Test driver generation
- Requirement-to-test traceability

Model captures component properties

The screenshot displays a P&ID diagram titled "P&ID Diagram: CHL1, 15. huhtikuuta 2013, 10:39". The diagram includes components like Tank (Tk1), Pump (CP), Pressure Indicator (PT1), Valves (V1, V2), and Heat Exchangers (Hx1, Hx2). Three property dialog boxes are open:

- Valve: Object** (Name: V1):
 - Number of valve ends: Two
 - Valve type: Normal, Manual, Control
 - Closing by: ball, butterfly, diaphragm, needle
 - Flow rate: 1200
 - Flow rate units: gpm
- Heat Exchanger: Object** (Name: Hx1):
 - Flow rate: 11000
 - Flow rate units: gpm
 - Pressure: 200
 - Pressure units: psig
 - Temperature: 200
 - Temperature scale: F
- Pipe: Relationship**:
 - Diameter: 10
 - Length: 10
 - Cover: none

A text box on the left states: "Potential Flow rate Issue (seeded defect)". A blue arrow points from this text to the "Flow rate" field in the "Valve: Object" dialog, which is highlighted with a red box. A red arrow also points from this field to the "Flow rate" field in the "Heat Exchanger: Object" dialog, which is also highlighted with a red box.

Analysis identifies unsatisfiable constraints

t_150 Coverage Analysis **FAILED**

Vector File: C:\Users\jpt\Documents\MetaEdit+ 5.1\reports\test_vectors\t_150.TST

Time Run	02-29-16 17:37:27
Analyzer Version	4.0.0
Total Number of DCPs	2
DCPs Not Covered	1
Predicates Requiring Coverage	11
Predicates Not Covered	5 cv_t_150_RP_1 t_150 t_150_FR_1 t_150_RP_1 t_150_1_LS
Total Coverage Errors	6
Total Coverage Warnings	0
Test Generation Failures	2

TTM Table t_150

#	Assignment	Condition	Requirement ID
1	flow_rate = Hx1.flow_rate; pressure = Hx1.pressure; temperature = Hx1.temperature;	Hx1.flow_rate = 11000 AND Hx1.pressure = 200 AND Hx1.temperature = 200 AND t_V1.flow_rate = Hx1.flow_rate AND t_V1.pressure = Hx1.pressure AND t_V1.temperature = Hx1.temperature	Req_Hx1_468: Objec...
2	flow_rate = Hx1.flow_rate; pressure = Hx1.pressure; temperature = Hx1.temperature;	Hx1.flow_rate <= 11000 AND Hx1.pressure <= 200 AND Hx1.temperature <= 200 AND t_V1.flow_rate = Hx1.flow_rate AND t_V1.pressure = Hx1.pressure AND t_V1.temperature = Hx1.temperature	Req_Hx1_468: Objec...

Hyperlink from analysis report to model defect

Uncovered DCP Paths

DCP Number	DCP Path	Failure Detection
1	t_150, t_150_FR_1, cv_t_150_RP_1, t_150_RP_1, t_150_RP_0, t_150_1_LS (goto model)	Vector Generator

Failed Pre-Condition Relation	cv_equal_to
Exact SS File Location	t_150.SS Line # 30
Input Domain At Error	Occurance at UA = 662 t_V1_VAR.flow_rate:Valve_type_flow_rateDataType [0 .. 1200] Hx1.flow_rate:Heat_Exchanger_type_flow_rateDataType [11000 .. 11000]

Experiences from the CHL case

- Model-based approach to:
 - Analyze requirements and create design specifications
 - Generate implementation code, deployment, docs etc.
 - Generate tests and provide traceability of tests to corresponding requirements
- Benefits
 - Improved system validation
 - Ability to better trace rationale
 - Improved systems engineering

Why DSM: emerging and enabling technology for V&V

- Provides relevant and intuitive graphical abstraction for specific domain or related subdomains
- Allows for rich semantics required for formal analysis and test generation
 - Necessary for V&V effectiveness and efficiency
- DSM tooling allows multiple views to be integrated
 - Model transformation often built into the tools
 - Integrates formal analysis and test generation tools
 - Formal methods hidden behind the scenes
 - Model languages are evolvable

Summary

- DSM provides differing views of system designs across multiple disciplines
 - Approach enables collaboration between domain experts and developers, development and verification
 - Leverage and integrate new analysis and model-based automation (e.g., simulation, synthesis, generation, etc.)
- Future challenges – increase in modeling & analysis tools
 - Apply further views and multi-model consistency (e.g. in automotive)
 - Integrate with various testing tools, each focusing on different aspects



MetaCase

Thank you!
Questions, please?

For references on examples and cases contact:
Juha-Pekka Tolvanen, jpt@metacase.com
www.metacase.com

References

- Blackburn, M., Denno, P., Virtual Design and Verification of Cyber-Physical Systems: Industrial Process Plant Design, Procedia Computer Science 28, Elsevier, 2014
- Kelly, S., Tolvanen, J.-P., Domain-Specific Modeling: Enabling Full Code Generation, Wiley, 2008. <http://dsmbook.com>
- EADS, www.metacase.com/papers/MetaEdit_in_EADS.pdf
- Elektrobit, O.-P. Puolitaival et al, Utilizing Domain-Specific Modeling for Software Testing, Proceedings of VALID, October 2011
- NSN, Architecture in the language, www.metacase.com/cases/architectureDSMatNSN.html
- Nokia, www.metacase.com/papers/MetaEdit_in_Nokia.pdf
- Panasonic, Proceedings of Domain-Specific Modeling, 2007, www.dsmforum.org/events/DSM07/papers/safa.pdf
- Polar, Proceedings of Domain-Specific Modeling , 2009, www.dsmforum.org/events/DSM09/Papers/Karna.pdf
- USAF, ICSE, <http://dl.acm.org/citation.cfm?id=227842>

